03｜工具选择：什么工具框架值得用？

2022-03-25 柳胜

《自动化测试高手课》 课程介绍 >

讲述：柳胜

时长 17:08 大小 15.69M

你好，我是柳胜。

工具选型评选会你应该不陌生，无论你是作为评审者，还是方案建议人。不过常常出现的场景就是，方案建议人讲了一通新工具 A 如何优秀强大，从问题分析到方案解决一应俱全。但参会专家并不是全都熟悉这个新工具，就会问出一堆类似“为什么你不用工具 B”的问题。

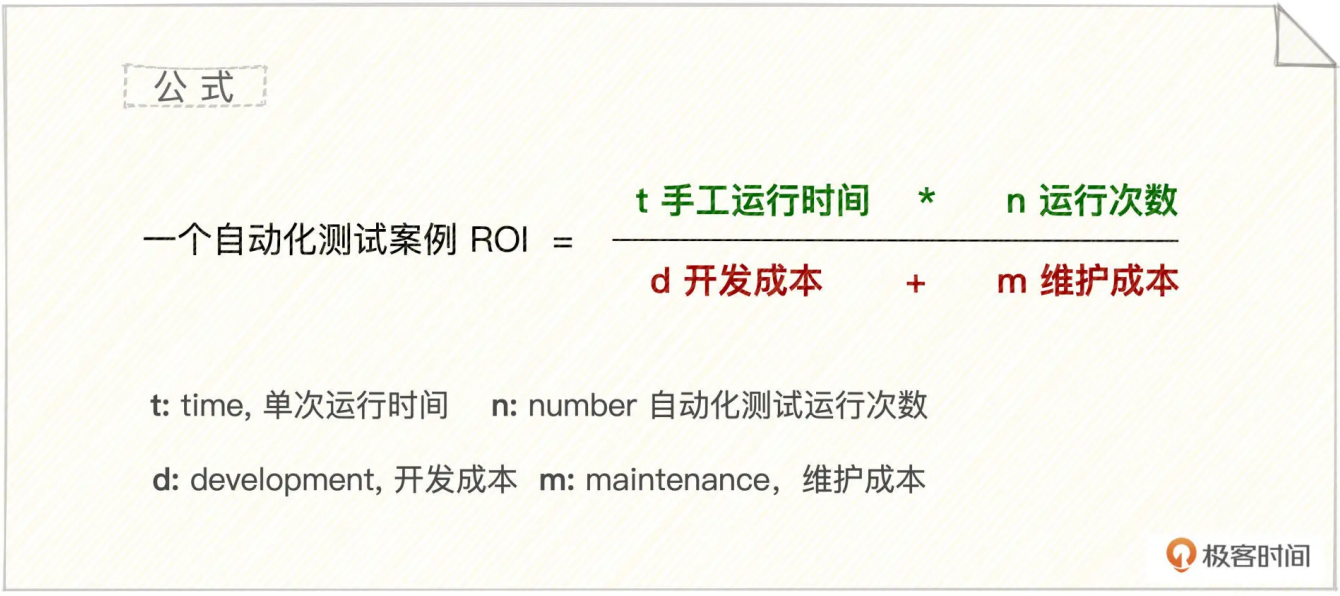
结果也可想而知，懂 A 的不懂 B，懂 B 的不懂 C，懂 C 的不懂 A，最后评审会咋决策？只好陷入一个依靠个人影响力来决策的模式。要破除这种死循环，就要找到一个量化模型。在我看来，量化数据才是团队有效交流的手段，也是团队达成共识的基础。

有了这样的模型，好处多多。评审阶段，可以更科学地预估某个工具的能力和风险；工具投入使用后，模型依旧能帮你持续观测，检验这个工具是否带来了预期价值。

今天我会带你一起推演出这个模型。学会推演方法，远比套用“模型”更重要。

自动化测试案例的成本

在开始量化评估之前，我们先回顾一下前面学过的 ROI 模型。



通过这个模型，我们得到一个重要结论：**一个自动化测试案例的开发工作量，在给定条件下，什么经验的工程师用什么工具，需要多长时间完成，这是可以估算的定值。但维护工作量包含了多种可变因素，是自动化测试项目的风险所在。**

今天我们聚焦公式里的分母，也就是开发成本 d 和维护成本 m。

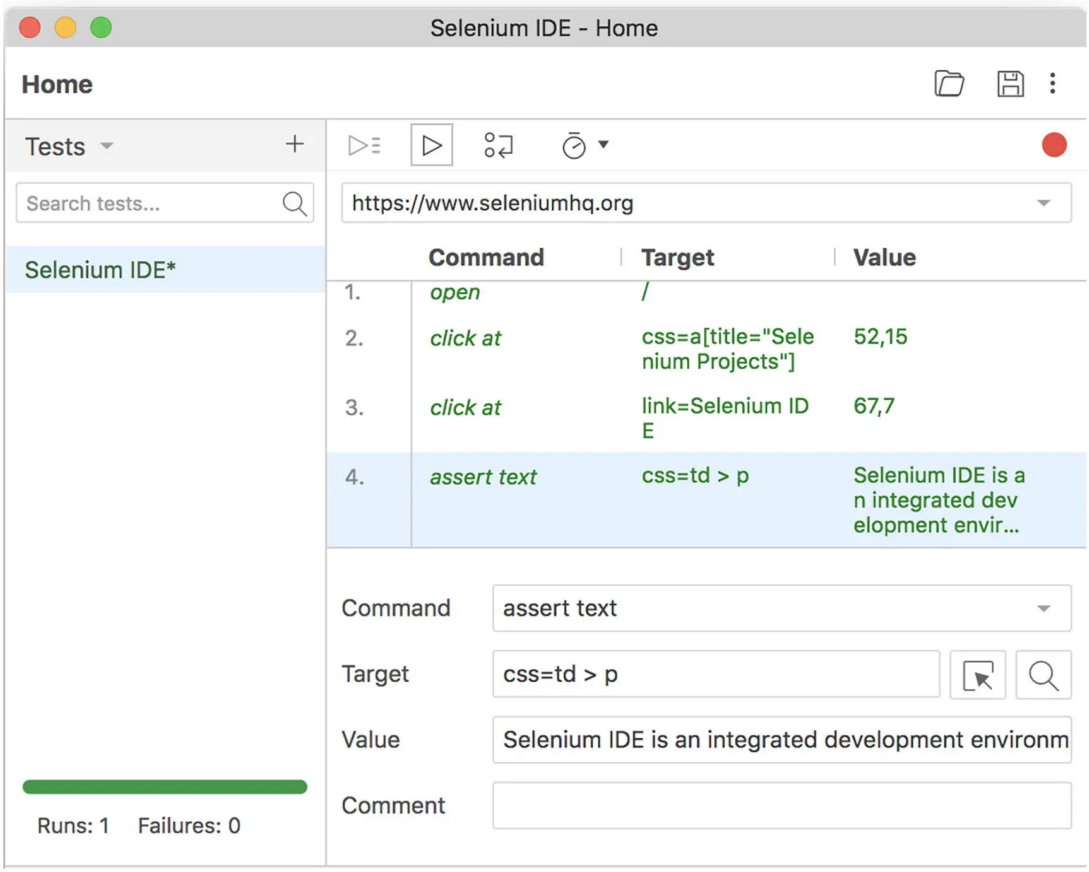
先说开发成本。在软件开发中，估算开发规模的传统方法有功能点和代码行，前沿的方法也有用 AST 抽象语法树。那如何估算开发一个自动化案例需要多大工作量呢？

首先我们要看一下自动化测试案例是如何产生的，目前在业界，主要有 5 种方式。

方法一：录制和回放方法

第一代的自动化测试工具大多基于录制回放，像最早的 Winrunner 就是录制桌面 UI 应用的。目前代表工具就是 selenium IDE。

要生成测试代码很简单，开启浏览器 selenium 的插件，打开测试的网页，比如 http://www.baidu.com，输入关键字“极客时间”，点击搜索，就会自动生成测试脚本了，如下图。



自动生成测试脚本演示

通过录制产生自动化脚本的这种方法，优点是速度快、零编码，对测试人员技术要求低。缺点是规模一旦扩大，维护工作量几乎无法承受，比如和 CICD 集成、自定制报告、多环境支持等等。

方法二：关键字编程

不过，录制回放产生的脚本，还是面向过程的一个个函数，还需要测试人员有一定代码基础，才能扩展和维护这些函数。

那么，有没有办法，让没有代码经验的人也能编辑、维护脚本呢？关键字驱动方式应运而生了，它增加了页面控件对象的概念，调用对象的方法就是操作对象运行，在这种机制下，对象、对象的行为、输入的数据和描述信息，这些内容都能用一个表格的形式呈现。业务人员只需要编辑表格，就能修改运行逻辑了，这就叫做**关键字驱动**。

下面是一张关键字驱动表格，编程就是编辑表格里的关键字和对象。关键字是一组预定义好的指令，编辑完表格后，框架会驱动这个表格，执行设定好的命令，来完成自动化测试。

关键字序号	描述信息	关键字	对象
第1步	在 home 页面点击“登录”链接	clicklink	//*[@id="menu-main"]/li[3]/a
第2步	输入用户名	inputdata	id="username"
第3步	输入密码	inputdata	id="password"
第4步	点击登录按钮	clickbutton	class="loginBtn"

极客时间

关键字驱动表格示例

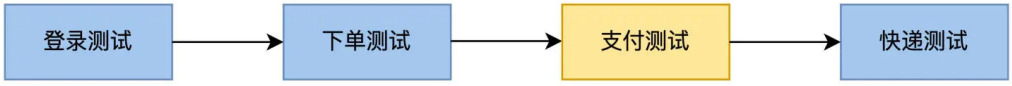
相比录制回放，关键字驱动框架的优势在于降低了测试开发人员的技术要求。而且测试开发人员对代码还有了更多的逻辑控制能力，比如增加循环结构、wait time、log 输出，只要框架提供足够丰富的关键字就行。

但你也不难看出，编辑表格的人和维护关键字仓库的人，并不是同一拨人。前者是对业务了解的测试人员，后者是技术能力强的开发人员，这样开发维护起来会增加难度。

方法三：模块库开发

随着软件技术的发展，自动化测试人员的技术水平也在提高，要解决的问题也更加复杂。比如自动化测试的代码怎么能够有效地复用，有没有好的扩展能力等等。这个扩展能力是二维的，分为水平功能扩展和垂直层级扩展。

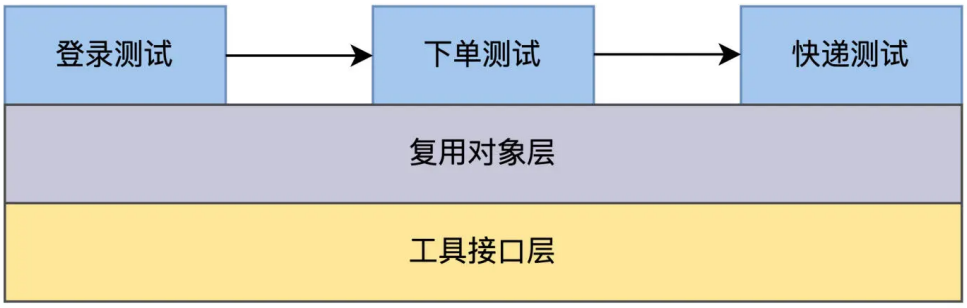
水平功能扩展指的是测试功能增多，自动化测试代码就借鉴了软件的模块设计思维，一个应用的测试场景可以切分成多个功能模块。比如订餐的流程可以分成登录模块、下单模块、快递模块，模块之间通过调用关系连接起来，组成测试场景。



极客时间

订餐流程模块示意图

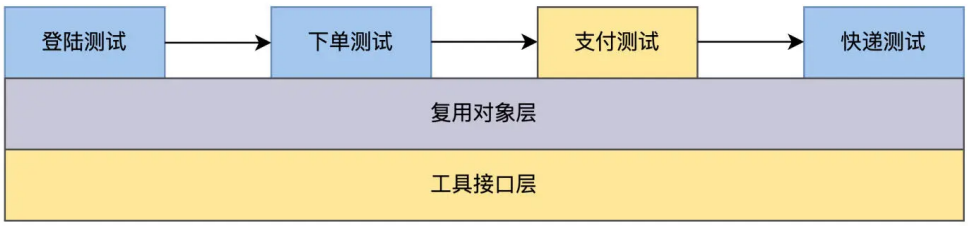
而在技术层面上，又可以垂直切分出功能案例库和通用库。比如页面的组件可以形成复用库、page 对象、button 对象、link 对象等等，把和开发技术耦合的技术层封装在复用库里，而和测试相关的业务功能实现在功能案例库里。



极客时间

订餐流程垂直切分示意图

这样的设计，遵循了高内聚低耦合的软件设计思想，未来自动化测试规模扩展时也很方便。比方说增加一个支付功能，就可以把支付页面的对象写到复用库里，新创建一个支付测试案例，前面和下单模块衔接，后面和快递模块对接，就能跑起来了。



极客时间

模块扩展示意图

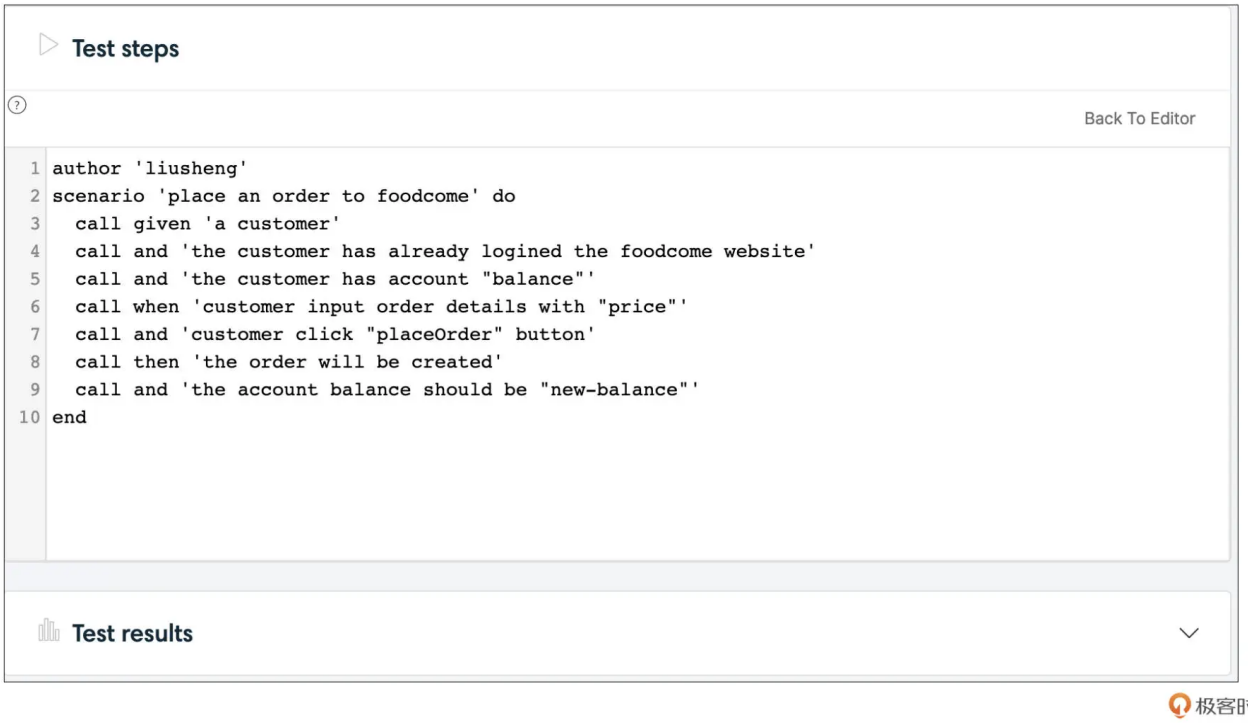
模块库开发的优点是可扩展、可复用，困难是需要架构工作和代码工作，对自动化测试开发人员的代码能力要求也比较高，接近开发工程师。

方法四：BDD 混合框架

还有一种方法是 BDD 混合框架。BDD 全称是 Behavior Drive Development，行为驱动开发，它通过 Gherkin 语法定义测试场景。

Gherkin 语法包含一套类自然语言的关键字：when、given、then，given 描述条件，when 描述行为，then 描述结果。这样一个场景的 3 要素：上下文、动作和结果就说明白了。

所以，Gherkin 语法描述出来的测试场景，能够同时被非技术和技术人员理解。客户、需求人员、开发人员和测试人员从 BDD 案例各取所需，需求人员得到用户手册，开发人员得到 Use Case，测试人员得到测试案例。这些都有 BDD 框架支持生成代码，比如 Cucumber。



BDD方法示意图

BDD 和关键字驱动框架的优缺点基本一样，不同的是，BDD 不局限于测试框架，能够和软件流程集成在一起，也有相应的开发 IDE 插件，比如 Eclipse plugin。

方法五：更高 ROI 的探索，自动化前沿技术

最后，我要说一下目前比较火，听起来也很酷的自动化前沿技术。

AI 测试曾被寄予厚望，我们期待由它发展出一套自动化测试全栈解决方案，可以自动生成测试案例和代码，而且维护工作量为零。不能说这些全都是镜花水月，我相信目前 AI 只在一小部分测试领域里落地，比如图像识别的 Applitool，可以用作图片验证；游戏领域里的监督学习，用来行为克隆等等。

我也曾看过一些 AI 根据规则自动生成测试案例的演示，但演示只是演示，它演示的方案需要的很多条件，现实还不具备，比如基于非常理想的数据模型等等。所以，我认为 AI“落地”的定义是，它的形式是产品，而不是个人业余的项目或者一段开源代码。

相比 AI 测试，我更看好另外一种自动化生成测试的思路，就是基于规则化或可以模式化的业务场景，把案例的生成和代码生成一并自动化，形成一种可以量化的案例发现方案。我会在第 5 讲带你了解这个思路如何实现。

记住，**测试工作是要能证明软件功能的成败，其方法论基石是确定论，而不是未知论。**说得通俗一点，测试是在编网，虽然网会漏鱼，但我很确信只要投入人手和时间，就能把网编到什么程度，网住什么鱼。 而不是今天我捉到一条鱼，明天不知道鱼在哪里。这是我不认为 AI 能完全替代手工测试工作的原因。

好，业界已知的生成自动化测试方法，我们做了归类了解，接下来再分析不同类型的自动化测试开发成本和维护成本。

我给这个表格起一个名字，叫做**案例 DM 分析表**。D 代表 Development，M 代表 mantainence。

案例生成方法	业界工具	案例开发成本	案例维护成本	最佳实践
录制和回放	Web: Selenium IDE, Windows:WinApp Driver Desktop: TestStudio,Sikuli	低	高	手工团队 自动化规模小
关键字驱动	QTP	中	高	初建自动化团队 自动化规模中
模块库开发	Simulato XUnit	高	低	有经验团队 自动化规模大
BDD 混合	Cucumber SpecFlow Guage	中	中	跨团队协作 自动化规模中

极客时间

工具的成本

前面探讨的给定了类型工具的情况下，案例的开发、维护成本如何衡量。但别忘了，成本计算时，还有个不小的成本：工具和框架的成本。这个在选型阶段没有考虑的话，会给后面埋下很多隐患，带来不少让人头疼的问题，比如下面这些。

1. 当你的团队开始上手案例开发时，发现框架只支持 JavaScript，Perl，但你的团队都是熟悉 python 和 Java。在这种情况下，是培训已有团队还是招聘新的人才，还是弃用方案而选择新的框架？

2. 当你需要升级框架时，高版本对低版本的兼容性。它有可能让你的原有测试案例不能工作，最糟糕的是，会出现一些奇奇怪怪问题，每一个问题的诊断过程是一场对你毅力和耐心的考验。

3. 当你的案例在运行的时候，遇到一个框架抛出来的 exception，网上搜不到解决方案，去社区论坛提问，也没人响应，自己去看源代码又陷入浩若烟海的 context 中去，那种无力感蔓延开来，直让你怀疑人生。

4. 当你的案例规模扩展，新的 API 案例需要调用 PATCH 原语，但你已使用了一年的 API 框架不支持 PATCH，更悲催的是，你发现这个 API 框架已经停止更新，凋零的社区、隐身的支持人员、破败的代码，都是对你的毒打。

进坑容易出坑难。我见证过一个自动化测试团队选型工具草率，导致在 2 年内更换了 4 个工具。每次换工具，都会重写一遍自动化测试案例，人收获了教训离开公司另谋高就，但公司浪费了 2 年的时间和资源，之后还要换一拨人乐此不疲地重复这样的故事。

可以说，选型合适的框架是自动化测试架构设计中一个重要的选择，它通过开发成本和维护成本两个因子影响自动化测试 ROI，ROI 低于 1 的时候，这个项目就要 over 了。

那什么是 ROI 好的框架呢？

首先，框架要**满足我们的测试需求**。比如，UI 框架能有对象识别能力，API 框架能有 http 原语封装，对 xml 和 json 的支持，单元测试框架能有 mock 能力。

其次，框架应该有**广泛的同行用户、持续的更新、成熟的社区和积极的客户响应**，这些也能帮我们降低维护框架的成本，获得更好的 ROI。

把这些因素量化，就能得到一个工具四维成熟度表，你可以把它作为原始模型，用到你的工作里。

度量名字	度量指标例子	谁来度量	什么时候度量
语言的人气	Java, C#, JavaScript	自动化测试架构小组	预研阶段
用户的数量	1 万用户	自动化测试架构小组	预研阶段
更新的频率	n 个版本/每年	自动化测试架构小组	预研阶段，运行阶段
社区成熟度	活跃程度，响应时间	自动化测试架构小组	预研阶段

极客时间

工具四维成熟度表

小结

今天这一讲，我们从开发成本和维护成本的角度系统梳理了工具和框架的优劣势。根据脚本的生成方式的不同，可以划分出录制回放工具、关键字驱动工具、模块库开发工具、BDD 混合工具和 AI 工具，针对自动化的不同类型和规模，脚本的开发和维护成本也会发生变化。

框架本身也会有维护成本，这里我们优先选择成熟主流的框架，会降低维护的成本。这里“成熟主流”，我们使用用户的数量，更新的频率，社区成熟度，开发语言与团队的适配度四个指标来对比度量。

我并没有给出一个精准计算工具框架 ROI 的公式，而是给出几个维度来帮你厘清工具和框架的选择思路。

记住，这里有两大原则：

1. 按照团队的**技术能力、项目的预期长短、将来扩展的规模大小**，选择 ROI 最大的工具框架；

2. 当工具框架带来的 ROI 无法升高时，就考虑按照原则 1 **重新评估**，选择 ROI 更好的工具框架。

所以，框架和工具的评估不是一劳永逸，而是在整个自动化测试生命周期内实时观测，如果有变要及时止损，让 ROI 持续提升。

思考题

在工作中，你的团队非常熟悉 Python，所以选择某个支持 Python 框架。基于这样的原因做选择是不是遵循 ROI 原则？这样的选择会带来什么样的好处和坏处？

欢迎你在留言区跟我交流互动，也推荐你把这一讲分享给更多同事和朋友，说不定就能帮他通过下一次的工具选型评审会。

分享给需要的人，Ta订阅超级会员，你最高得 50 元

Ta单独购买本课程，你将得 20 元

生成海报并分享

赞 2 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 02 | 3KU法则：如何找出最优自动化实施截面？

精选留言 (3)

写留言

太匆匆

2022-03-25

录制回放和关键字驱动目前仍然是使用最频繁的方法，模块化开发感觉是关键字驱动的更高的阶段，BDD不仅限于测试了。很期待第五讲了

作者回复: 总结的不错！

2

swordman
2022-03-26

几年前，我所在的研发团队，在移动端产品中，在Appium框架上实现了关键字驱动，只是我们没有用表格，用的是YAML脚本。正因为编辑脚本和维护关键字不是同一批人，沟通与维护工作量较大，导致后来测试用例没有形成规模，这个方案没有达到预期的效果。 老师提到的思考题，我觉得优点是团队可以用Python快速开发出测试用例，缺点是没有明确这个Python框架需要满足什么测试需求，比如selenium和appium都支持Python，可以满足...
展开

作者回复: 你的名字很cool！思考得也很到位！无论什么工作，分工越多，越会增加团队内hand over信息的次数，这个协作和沟通的成本让团队失去活力，整天忙着开会，做报告。高效团队最关键的是人的热情和成就感！

IT蜗壳-Tango
2022-03-25

打卡

编辑回复: 都追到新一节课了，赞，期待周一更新吧，后续更有料哦~